Pr/T네트를 이용한 규칙베이스의 일관성과 완전성 검사

Consistency and Completeness Checking of Rule Bases Using Pr/T Nets

조 상 엽*
Sang-Yeop Cho

요 약

기존의 규칙베이스 검증 알고리즘은 명제논리 수준에 대응하는 지식표현에 대한 것들이었다. 실세계에서 규칙베이스를 구성할 때는 술어논리 수준에 대응하는 규칙을 사용하여 규칙베이스를 구성한다. 그러므로 본 논문에서는 규칙을 술어논리 수준에서 자연스럽게 표현할 수 있는 Pr/T 네트를 이용한 규칙베이스의 검증 알고리즘을 제안한다.

Abstract

The conventional procedure to verify rule bases are corresponding to the propositional logic-level knowledge representation. Building knowledge bases, in real applications, we utilize the predicate logic-level rules. In this paper, we present a verification algorithm of rule bases using Pr/T nets which represent the predicate logic-level rules naturally.

1. 서 론

규칙기반 표현 방법은 규칙의 모듈성과 비절차적인 지식처리과정 때문에 전문가시스템에서 지식을 표현하는데 가장 많이 사용하는 방법이다. 이러한 장점에도 불구하고 규칙기반 전문가시스템의 추론과정이 사용자, 개발자 그리고 전문가에게 보이지 않는다는 문제점을 가지고 있다. 그리고 전문가시스템이 일정한 기간동안 개발됨으로 일관성이 없는 개발 요구서에 의해 점진적으로 개발된다. 이러한 결과, 상업적으로나 또는 인명과 재산의 피해가 우려되는 곳에 사용되기 전에 전문가시스템이 가지고 있는 오류를 제거하기 위한 규칙베이스의 검사를 사전에 실시하여야한다[1].

학교, 회사 등의 기관뿐 아니라 일반 가정에까지 보급된 컴퓨터와 인터넷은 정보의 교환 및 습득에 큰 영향을 미치고 있다. 이에 따라 컴퓨터 발명 초기의 수치와 통계 처리에서 최근에는 자연언어로 표현된 문장의 구조와 의미를 처리함으로써 언어 지식적이고 추상적이며 차원 높은 정보를 획득하려는 방향으로 연구가 진행되고 있다[1].

규칙베이스를 검증하기 위한 많은 방법들이 개발되고 제안되었다. 초기의 규칙베이스의 검사는 규칙베이스에 존재하는 규칙을 한 쌍씩 검사하여 오류를 찾아내는 방법들이 소개되었다[2]. 최근에는 한 쌍의 규칙에서 발생하는 오류보다는 한 쌍의 추론사슬 상에서 발생하는 오류를 찾아내는 방법들이 연구되고 있다. 이러한 기법들은 규칙베이스에 존재하는 규칙들에 대응하는 그래프 구조로 변환하고, 그래프 구조의 위상적인 속성인 도달문제로 변환하여 규칙베이스내의 오류를 검사한다.

이러한 그래프의 속성을 이용한 검사방법에 사용하는 그래프로는 방향성 그래프(Directed Graph), 페트리네트(Petri Net), 추론그래프(Inference Graph), 초월그래프(Hypergraph), 방향성 초월그래프(Directed Hypergraph) 등이 있다. 이러한 그래프를 사용한

* 종신회원 : 청운대학교 인터넷컴퓨터학과 교수
sycho@chungwoon.ac.kr

방법은 규칙베이스의 규칙을 명제논리 수준의 그래프표현으로 변환하여 검증하는 방법이다[3-9].

그러나 실세계에서 사용하는 규칙베이스의 규칙은 술어논리에 기반을 둔 지식표현방법을 사용한다. 그러므로 그래프를 이용한 검증방법도 규칙베이스의 규칙을 명제논리 수준이 아닌 술어논리 수준으로 표현하는 그래프를 이용하여 검증절차를 개발하는 것이 타당하다.

본 논문에서는 규칙베이스의 규칙을 명제논리 수준이 아닌 술어논리수준의 지식을 표현할 수 있는 그래프 표현으로 변환하여 규칙베이스를 검사하는 방법을 제안한다. 규칙을 술어논리 수준이 그래프 표현으로 변환하기 위해 페트리네트의 술어논리수준의 확장인 Pr/T네트를 사용한다[10-12]. Pr/T네트를 이용하여 규칙베이스의 규칙들을 그래프로 표현하고 그래프에 대응하는 투사행렬로 변환하여 행렬연산을 통하여 규칙베이스내의 오류를 찾아낸다.

이러한 절차에 사용하는 규칙베이스의 규칙들은 단일한 형식(unitized form)을 갖는다. 단일한 형식은 규칙의 전제부는 논리곱표현(conjunction)만을 허용하고 결론부는 단지 단순절(simple clause)만을 허용하는 형식이다. 이러한 형식으로 구성된 규칙베이스는 Paderson이 기술한 것처럼 규칙베이스의 일반성을 잃지 않는다[13-15].

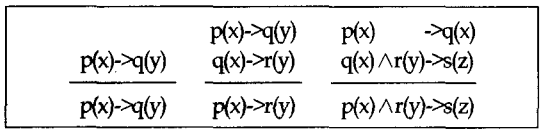
본 논문의 구성은 다음과 같다. 제2장에서는 검증 알고리즘이 검사할 일관성과 완전성에 대해서 설명한다. 제3장에서는 Pr/T 네트에 대하여 설명한다. 제4장 Pr/T 네트의 속성을 이용하여 검증하는 방법을 제안한다. 마지막으로 제5장에서는 결론을 기술한다.

2. 일관성과 완전성

전문가시스템의 규칙베이스에 존재하는 구조적인 오류를 찾아내는 과정을 검증(verification)이라고 한다. 규칙베이스의 검증은 일관성(consistency) 검증과 완전성(completeness) 검증으로 구분한다. 일관성 검증은 규칙베이스의 중복(redundancy), 포함(subsumption), 충돌(conflict), 순환(circularity) 등을 그리고 완전성 검증은 단절(deadend), 비도달(unreachable) 등과 같은 구조적인 오류를 찾아내는 것이다[5,6,8].

2.1 중복

중복은 두 개 이상의 서로 다른 규칙의 집합이 주어진 같은 값에 대하여 같은 결론을 유도할 때 발생한다. 중복은 시스템에 치명적이지는 않지만 시스템 성능의 저하를 가져온다. 그러나 시스템을 점진적으로 개발하면서 한 규칙의 집합은 수정하였으나 다른 규칙의 집합을 같이 수정하거나 또는 제거하지 않으면 이들은 충돌과 같은 치명적인 오류를 발생할 잠재적인 가능성이 있다. 중복은 다음과 같은 경우에 발생한다.



(그림 1) 중복의 예

중복된 규칙의 집합들 중 한 규칙집합을 제거할 때는 추론통로를 고려하여야 한다. 예를 들어 다음 세 개의 규칙을 생각해보자.

규칙1 : p(x) -> q(y)
규칙2 : p(x) -> r(z)
규칙3 : r(z) -> q(y)

규칙집합 2와 3은 규칙1과 같은 결론을 유도한다. 중복이 발생함으로 규칙집합 2와 3 그리고 규칙3 중에 한 규칙의 집합이 제거되어야 한다. 만일 규칙집합 2와 3이 단지 q(y)를 유도하기 위한 것이라면 이 규칙집합이 중복이 된다. 그러나 규칙집합 2와 3이 다른 추론통로를 설정하기 위해 필요한 것이라면 규칙1이 중복이다. 그리고 한 절(clause)에서 다른 절까지 복수 개의 통로가 있다고 하여

반드시 중복이 되는 것은 아니다. 즉, 추론통로의 중간 절이 다른 절과의 추론통로를 구축하기 위한 것이라면 중복이 발생하지 않을 수 있다. 그리고 x, y, z가 상수가 아니고 전부 또는 일부가 변수라면 잠재적인 중복이 된다.

2.2 포함

포함은 두 개 이상의 서로 다른 규칙집합이 같은 결론을 갖지만 한 규칙집합의 첫 번째 규칙의 전제부가 더 많은 제약 조건을 가질 때 발생한다. 즉, 한 규칙의 전제부가 다른 규칙의 부분집합이 될 때 발생한다. 포함은 중복과 마찬가지로 치명적인 오류는 아니지만 동시에 수정되지 않으면 다른 오류를 발생시키는 원인이 되므로 규칙베이스에서 제거하는 것이 바람직하다.

만일 포함 중에서 한 개의 규칙만을 갱신하게 되면, 규칙베이스 내에 또 다른 오류를 발생시킬 수 있고, 포함규칙집합의 갱신을 주의 깊게 하지 않으면 규칙베이스에 틈(gap)이 발생하거나, 특정한 추론을 유도할 수 없게 된다. 즉, 포함은 그 자체가 문제를 직접적으로 야기시키기 보다는 시스템을 개발하고 유지 보수하는 지식공학자를 괴롭히는 문제이다.

규칙베이스에 존재하는 포함규칙을 해결하는 방법이 몇 가지가 있다. 첫째는 전제부가 더 자세히 기술된 규칙을 제거하거나, 둘째는 전제부가 더 일반적으로 기술된 규칙을 제거하는 것이다. 셋째는 전문가시스템이 사용하고 있는 충돌해결(conflict resolution) 전략을 이용하여 규칙을 추론 때 선택하는 방법이다. 그리고 규칙에 우선순위를 부여하여 추론 시에 실행될 규칙을 제어하는 방법이다.

포함은 다음과 같은 경우에 발생한다.

p(x)∧q(y)->r(z)	p(x) ->q(y) q(x)∧r(y)->s(z)	p(x)∧q(y)->r(z) r(x)∧s(y)->t(z) t(x)∧u(y)->v(z)
p(x) ->r(z)	p(x) ->s(y)	p(x)∧u(y)->v(z)

(그림 2) 포함의 예

2.3 충돌

전문가시스템에 입력으로 같은 사실(fact)이 주어지면 일관된 추론 결과가 나와야 한다. 그러나 서로 다른 결과를 유도하는 상황이 발생할 수가 있다. 이와 같은 충돌은 같은 전제부가 상호 배타적인 결론을 유도할 때 발생한다. 충돌이 발생하는 상황은 같은 전제부에 대해서, 결론이 논리적 부정관계라기 보다는 상호배타적(mutually exclusive)인 관계일 때 발생한다고 보는 것이 보다 타당하다. 다음 기술에서 p와 ~p는 논리적 부정관계가 아니고 상호배타적인 관계를 의미한다.

충돌은 상업적인 시스템 또는 인명과 재산에 많은 영향을 주는 곳에서 운영하는 시스템에서는 반드시 제거되어야할 오류이다.

충돌은 다음과 같은 경우에 발생한다. x, y, z가 상수가 아니고 전부 또는 일부가 변수라면 잠재적인 충돌이 된다.

p(x)->q(y)	p(x)->q(y) q(x)->r(y)	p(x) ->q(y) q(x)∧r(y)->s(z)
p(x)->~q(y)	p(x)->~r(y)	p(x)∧r(y)->~s(z)

(그림 3) 충돌의 예

2.4 순환

순환은 일련의 규칙의 추론결과가 다시 추론사슬의 첫 번째 규칙의 전제부 또는 전제부의 일부분을 유도할 때 발생한다. 이것은 속성들간의 인과관계의 방향을 식별하기가 쉽지 않을 때 발생한다. 순환이 발생하면 시스템이 고장나게 된다. 그러므로 순환은 규칙베이스에서 제거되어야한다.

순환은 다음과 같은 경우에 발생한다.

p(x) -> q(y) q(x) -> r(y)	p(x) -> q(y) q(x) -> r(y)
r(x) -> p(y)	r(x) -> ~p(y)

(그림 4) 순환의 예

2.5 단절

단절은 규칙의 결론부가 목표가 아니면서 다른 규칙의 전제부 또는 전제부의 일부가 아닐 때 발생한다. 이것은 최종 목표를 유도할 때 이 규칙의 결론을 필요로 하는 다른 어떤 규칙이 규칙베이스에서 빠진 것을 의미한다. 단절은 시스템의 추론을 멈추게 함으로 전문가로부터 필요한 규칙을 제공받아야한다.

그림 5에서 p(x)와 q(x)는 시작노드 그리고 v(x)는 목표노드라고 하자. 이때 규칙1의 결론 r(x)는 단절이 된다.

2.6 비도달절

비도달절은 규칙의 전제부가 입력이 아니면서 다른 규칙의 결론부가 아닐 때 발생한다. 이것은 이 규칙이 해결하려는 문제와 전혀 관계가 없는 것이거나 이 규칙의 전제부 또는 전제부의 일부를 구축해주는 다른 규칙이 빠진 것을 의미한다. 필요한 규칙이 빠진 것이라면 이것을 전문가에게 제공받아야한다. 그림 5에서 규칙4의 전제부 u(x)는 비도달절이 된다.

규칙1 : p(x) -> r(x)
규칙2 : p(x) -> s(x)
규칙3 : p(x) ∧ q(x) -> t(x)
규칙4 : u(x) -> v(x)
규칙5 : t(x) -> v(x)

(그림 5) 단절과 비도달절의 예

3. Pr/T 네트

이장에서는 Pr/T네트의 정의와 관련 용어를 설명하고 Pr/T네트의 그래프 표현에 대해서 기술한다[10-12].

3.1 Pr/T네트의 정의

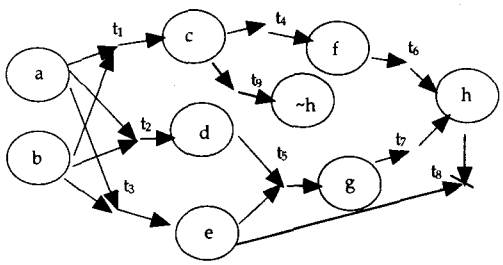
Pr/T네트는 명제논리수준에 대응하는 지식을 표현하는데 주로 사용하는 페트리네트에 대한 술어논리수준에 대응하는 확장된 지식표현방법이다. 다음은 Pr/T 네트의 정의와 정의에 필요한 용어와 정의 그리고 투사행렬에 대하여 설명한다.

- **상수(constant)** : U를 상수의 유한집합이라고 하자. U^k는 모든 상수 k-쌍 $<u_1, \cdots, u_k>$의 집합이다. 여기에서 $u_i \in U$이다.
- **변수(variable)** : V를 U상의 변수의 유한집합이라고 하자. V^k는 모든 변수 k-쌍 $<v_1, \cdots, v_k>$의 집합이다. 여기에서 $v_i \in V$이다.
- **항 집합(term set)** : W=U∪V라고 하자. W^k는 모든 k-쌍 $<w_1, \cdots, w_k>$의 집합이다. 여기에서 $w_i \in W$이다.
- **항의 수(arity)** : 만일 $x \in W^k$라면 x의 항의 수(arity)는 k가 되고 $\alpha(x)=k$라고 표기한다.
- **Pr/T네트의 정의** : PTN=(P, T, F, L)를 Pr/T네트라고 하자. 여기에서 P와 T는 각각 PTN의 술어(place로 표현)와 함축(transition으로 표현)의 유한집합이다. $P \cup T \neq \emptyset$, $P \cap T = \emptyset$. F는 흐름관계(flow relation), 즉 아크의 유한집합이고, F의 원소는 PTN의 아크이다. $F \subseteq (P \times T) \cup (T \times P)$. w(a,b)는 a에서 b로 가는 아크의 수이다. a와 b는 트랜지션 또는 플레이스이다. $domain(F) \cup codomain(F) = P \cup T$. $L: F \rightarrow W \cup \cdots \cup W^n$ 인 아크레이블함수이다. n은 P의 술어 p가 가질 수 있는 항(term)의 최대 수이다. $\forall p \in P$와 $\forall x, y \in \sum_{p \cap \{a,b\} \neq \emptyset} L(a,b)$에 대해서 $\alpha(x)=\alpha(y)$이다.
- **투사행렬** : PTN이 m개의 플레이스와 n개의 트랜지션을 갖는 Pr/T 네트라고 하자. PTN의 투사행렬(incidence matrix)은 n×(m+복합노드의 수) 정수행렬 $C=[c_{ij}]$. $c_{ij}=w(t_i,p_j)-w(p_j,t_i)$. t_i와 p_j는 각각 트랜지션과 플레이스이다.

3.2 Pr/T네트의 그래프 표현

규칙1 : a(x) ∧ b(y) → c(z)
규칙2 : a(x) ∧ b(y) → d(z)
규칙3 : a(x) ∧ b(y) → e(z)
규칙4 : c(x) → f(z)

규칙5 : d(x) ∧ e(y) → g(z)
규칙6 : f(x) → h(y)
규칙7 : g(x) → h(z)
규칙8 : e(x) → h(z)
규칙9 : c(x) → ~h(z)



(그림 6) 규칙에 대한 Pr/T 그래프

	a b	ab c	d e	de f	g h	~h
t_1	-1 -1	-1 1				
t_2	-1 -1	-1	1			
t_3	-1 -1	-1	1			
t_4		-1		1		
t_5			-1 -1	-1	1	
t_6				-1	1	
t_7					-1 1	
t_8			-1		1	
t_9		-1				1

(그림 7) 규칙에 대한 투사행렬

Pr/T네트를 이용하여 규칙베이스의 규칙을 술어논리수준으로 표현할 수가 있다. 규칙에서 나타나는 절은 플레이스로, 절에 나타나는 항은 아크로, 규칙은 트랜지션으로 각각 표현할 수 있다. 다음의 규칙들은 그림 6와 같은 Pr/T네트 그래프로 표현할 수 있고, 이 그래프에 대한 투사행렬은 그림 7과 같다.

ab와 de는 단순노드(simple node) a와 b 그리고 d와 e로 구성되는 복합노드(compound node)이다. 단순노드가 두 개 이상 모여 복합노드를 구성하고 입력 플레이스가 되면 복합노드에 -1을 준다.

3.3 행 갱신연산

Pr/T 네트 그래프에서 입력노드와 출력노드는 도달관계를 표시한다. 입력노드에서 도달한 각각의 출력노드가 복합노드를 구성할 때 투사행렬에 이러한 사항을 표시해야 한다. 즉, 복합노드에 의한 도달관계도 투사행렬에 표시되어야 만이 규칙집합간에 발생하는 일관성을 찾아 낼 수 있다. 이러한 관계를 찾기 위해 행 갱신연산을 수행한다. 행 갱신연산은 전제부가 같은 각 행에 대해 실행한다. 행 갱신연산 RowModify은 다음과 같다.

RowModify(M[T,V])
= M[T,V] + ∨i M[T,v_i], $\cup_i$ v_i=V, M[T,v_i]≠-1

여기에서 T는 전제부가 같은 행들이고, V는 복합노드이거나 단순노드이다. v는 단순노드이다. 그림 7에서 전제부가 같은 행은 t_1, t_2, t_3이다. 이들 중 출력노드가 복합노드를 구성하는 t_2와 t_3에 대해 도달관계를 표시하기 의해 de 열에 값이 갱신되어야 한다. 그림 8은 행 갱신연산을 실행한 후의 그림이다. 행 갱신의 결과는 t_2와 t_3의 de 열에 굵은 숫자로 표시하였다.

	a b	ab c	d e	de f	g h	~h
t_1	-1 -1	-1 1				
t_2	-1 -1	-1	1	**1**		
t_3	-1 -1	-1	1	**1**		
t_4		-1		1		
t_5			-1 -1	-1	1	
t_6				-1	1	
t_7					-1 1	
t_8			-1		1	
t_9		-1				1

(그림 8) 행 갱신연산 후의 투사행렬

4. 검증

4.1 검증 알고리즘

본 논문이 제안하는 검증 알고리즘은 초기 투사행렬 M_1을 가지고 시작한다. 알고리즘은 행렬간에 부호가 다른 행의 쌍을 조합하여 새로운 행렬

을 출력한다. 새로운 행렬과 기존의 행렬을 이용하여 일관성을 조사하는 것으로 검증을 실행한다.

검증 알고리즘

입력 : 초기 투사행렬 M_1, 시작노드 리스트, 목표노드 리스트
출력 : 검증결과
1. i=1
2. M_1와 M_i를 비교하여 일관성을 검사한다.
3. 투사행렬 M_1과 M_i에 부호가 다른 행의 쌍이 없을 때까지 다음을 실행한다.
 3.1 M_1과 M_i에서 부호가 다른 행의 쌍을 조합한다.
 3.2 조합한 결과 행을 투사행렬 M_{i+1}에 출력한다.
 3.3 for j=1 to i
  M_{i+1}과 M_j를 비교하여 일관성을 검사한다.
 3.4 i++

투사행렬간의 일관성 검사는 다음과 같이 한다.

- **중복** : 비교하는 두 행의 엔트리의 위치가 일치하면 중복이다.
- **포함** : 비교하는 두 행의 엔트리 중 1의 위치가 일치하고 한 행이 다른 행보다 -1을 더 많이 가지고 있으면 포함이다.
- **충돌** : 비교하는 두 행의 엔트리 중 -1의 위치가 일치하고 1의 위치가 상호배타적이면 충돌이다.
- **순환** : 투사행렬에서 조합을 한 결과가 0이 되면 순환이다.
- **단절** : M_1에서 열의 값이 1만 존재하고, 목표노드 리스트와 비교하여 목표노드가 아니면 단절이다.
- **비도달절** : M_1에서 열의 값이 -1만 존재하고, 시작노드 리스트와 비교하여 시작노드가 아니면 비도달절이다.

단절과 비도달절은 알고리즘이 실행되기 전에 초기 투사행렬을 조사하여 찾을 수 있다. 중복, 포함, 충돌, 순환은 알고리즘을 실행하여 찾을 수 있다.

4.2 검증의 예

그림 8을 초기 행렬 M_1으로 하여 검증을 실행하면 그림 9와 같은 투사행렬을 얻을 수 있다. M_1

M_1	a b	ab c	d e	de f	g h	~h
1	-1 -1	-1 1				
2	-1 -1	-1	1	1		
3	-1 -1	-1	1	1		
4		-1		1		
5			-1 -1	-1	1	
6				-1	1	
7					-1 1	
8			-1		1	
9		-1				1

(a) 투사행렬 M_1

M_2	a b	ab c	d e	de f	g h	~h
1,4	-1 -1	-1		1		
1,9	-1 -1	-1				1
(2,3),5	-1 -1	-1			1	
3,8	-1 -1	-1			1	
4,6		-1			1	
5,7			-1-1	-1	1	

(b) 투사행렬 M_2

M_3	a b	ab c	d e	de f	g h	~h
1,4,6	-1 -1	-1			1	
(2,3),5,7	-1 -1	-1			1	

(c) 투사행렬 M_3

(그림 9) 검증과정의 투사행렬들

	중복	포함	충돌
M_1:M_1	∅	∅	∅
M_1:M_2	∅	<(8):(5,7)>	<(9):(4,6)>
M_2:M_2	∅	∅	<(1,9):(3,8)>
M_1:M_3	∅	∅	∅
M_2:M_3	<(3,8):(1,4,6)> <(3,8):((2,3),5,7)>	∅	<(1,9):(1,4,6)> <(1,9):((2,3),5,7)>
M_3:M_3	∅	∅	<(1,4,6):((2,3),5,7)>

(그림 10) 일관성 조사 결과

의 1행과 4행의 c 열이 부호가 다른 행이 되므로 조합하여 M_2의 (1,4) 행을 구성한다. M_1의 2, 3 행과 5 행의 d, e, de 열의 부호가 다른 행이 되므로 조합하여 M_2의 ((2,3),5) 행을 구성한다. 여기에서 (2,3)은 출력 노드가 복합 노드를 구성하는 트랜지션들을 의미한다. 이와 같은 방법으로 M_2를 얻는다.

M_1과 M_2를 비교하여 일관성을 조사한다. M_1의 8

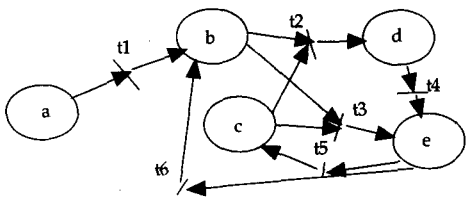
행과 M_2의 (5,7) 행은 1의 위치가 일치하고 (5,7) 행에 있는 -1의 수가 많기 때문에 포함이 된다. M_1의 9 행과 M_2의 (4,6) 행에 있는 -1의 위치가 일치하고 1의 위치는 상호배타적인 관계이므로 충돌이다. M_2내에서 (1,9) 행과 (3,8) 행은 -1의 위치가 일치하고 1의 위치가 상호배타적인 관계이므로 충돌이다.

M_1과 M_2를 부호가 다른 행의 쌍을 찾아 M_3를 생성한다. M_2의 (1,4) 행과 M_1의 6 행은 M_3의 (1,4,6) 행을 생성하고, M_2의 ((2,3),5) 행은 M_1의 7 행과 M_3의 ((2,3),5,7) 행을 생성한다.

M_1과 M_3에는 일관성에 문제가 없다. M_2와 M_3를 비교하여 일관성을 조사하면, M_2의 (1,9) 행과 M_3의 (1,4,6)과 ((2,3),5,7) 행은 충돌이 되고, M_2의 (3,8) 행과 M_3의 (1,4,6)과 ((2,3),5,7) 행은 중복이 된다. M_3내에서 (1,4,6) 행과 ((2,3),5,7) 행은 중복이 된다.

순환은 부호가 다른 행의 쌍을 조합한 결과가 0이 되면 발생한다. 그림 11은 순환이 발생하고 있는 Pr/T 네트 그래프를 보여준다. 그림 12의 M_1에서 5 행과 6 행의 bc 열은 행 갱신연산에 의해 1이 된다. M_2에서 (3,(5,6))과 ((5,6),3)은 조합한 결과가 0이 되므로 순환이 된다. M_3에서 (2,4,(5,6)), (4,(5,6),2), ((5,6),2,4)도 조합한 결과가 0이 되므로 순환이 된다. 즉 그림 11의 그래프에는 두 개의 순환이 존재한다.

단절과 비도달절은 초기 투사행렬을 조사하여 찾을 수 있다. 그림 13에서 시작노드는 a, b, ab이고 목표노드는 d와 g라고 가정하자. 노드 c는 목표노드가 아니면서 더 이상의 추론을 할 수 없기 때문에 단절이 된다. 노드 f는 시작노드가 아니면서 이 노드에 도달할 수 있는 방법이 없기 때문에 비도달절이 된다.



(그림 11) 순환이 나타나는 Pr/T그래프

M_1	a b	c bc	d e
1	-1 1		
2	-1	-1 -1	1
3	-1	-1 -1	1
4			-1 1
5		1 1	-1
6	1	1	-1

(a) 투사행렬 M_1

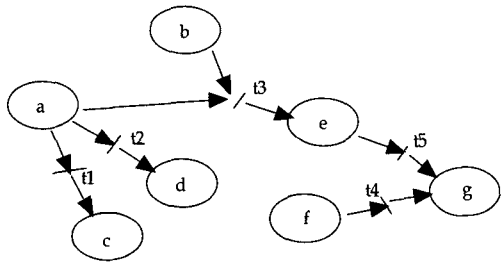
M_2	a b	c bc	d e
2,4	-1	-1 -1	1
3,(5,6)	0	0 0	
4,(5,6)	1	1 1	-1
(5,6),2			1 -1
(5,6),3			0

(b) 투사행렬 M_2

M_3	a b	c bc	d e
2,4,(5,6)	0	0 0	
4,(5,6),2			0
(5,6),2,4			0

(c) 투사행렬 M_3

(그림 12) 순환에 대한 투사행렬



(그림 13) 단절과 비도달절의 그래프

M_1	a b	ab c	d e	f g
1	-1	1		
2	-1		1	
3	-1 -1	-1	1	
4				-1 1
5			-1	1

(그림 14) 그림 13의 투사행렬

그림 4의 초기투사행렬을 조사하여 단절과 비도달 절을 찾아낸다. 열의 값으로 1만 존재하면 목표노드 또는 단절이 된다. 열의 값이 1만 존재

하는 노드는 {c, d, g}이고 목표노드는 {d, g}이므로 c는 단절이 된다. 열의 값으로 -1만 존재하면 시작노드 또는 비도달 절이 된다. 열의 값이 -1만 존재하는 노드는 {a, b, ab, f}이고 시작노드는 {a, b, ab}이므로 f는 비도달 절이 된다.

5. 결 론

본 논문에서는 Pr/T 네트의 속성을 이용하여 규칙베이스를 술어논리 수준에서 모형화하여 검증하는 방법을 제안하였다. 중복, 포함, 충돌, 순환 등의 일관성과 단절, 비도달절과 같은 완전성도 같이 찾아내는 검증 알고리즘을 제안하였다. 이 알고리즘은 기존의 알고리즘과 달리 규칙간의 일관성과 완전성뿐만 아니라 규칙 집합간에 발생하는 일관성과 완전성도 함께 찾아낸다. 복합절을 명시적으로 기술하여 복합절로 인하여 기존의 알고리즘에서 발생하는 불완전한 검증을 방지하였다.

참 고 문 헌

[1] Jackson, P. "Introduction to Expert Systems," Addison-Wesley, 2000.

[2] Gupta. U., Validating and Verifying Knowledge-based Systems, Readings, IEEE Computer Society Press, 1991.

[3] Agarwal, R., and Tanniru, M., "A Petri-net Based Approach for Verifying the Integrity of Production Systems," Int'l J. of Man-Machine Studies, Vol. 36, pp. 447-468, 1992.

[4] Nazareth, D. L., "Investigating the Applicability of Petri nets for Rule-based System Verification," IEEE Trans. KDE, Vol. 4, No. 3, pp. 402-415, 1993.

[5] Nazareth, D. L., and Kennedy, M. H., "Verification of Rule-based Knowledge Using Directed Graphs," Knowledge Acquisition, Vol. 3, pp. 339-360, 1991.

[6] Ramaswamy, M., Sarkar, S., and Chen Y., "Using Directed Hypergraphs to Verify Rule-Based Expert Systems," IEEE Trans. on KDE, Vol. 9, No. 2, March-April, pp. 221-237, 1997.

[7] Shiu, S. C. K., Liu, N. K., and Yeung, D. S., "An Approach Towards the Verification of Hybrid Rule/Frame-based Expert Systems Using Coloured Petri Nets," IEEE Int'l Conf. on SMC, pp. 2257-2262, 1995.

[8] 조상엽, "페트리네트를 이용한 규칙베이스의 검증," 한국정보처리학회 논문지, 제4권 제2호, pp. 430-440, 1997.

[9] Valiente, G., "Verification of Knowledge Based Redundancy and Subsumption Using Graph Transformations," Int'l J. Expert Systems, Vol. 6, No. 3, pp. 341-355, 1993.

[10] Genrich, H. J., and Lautenbach, K., "System Modeling with High-level Petri Nets," Theorectical Computer Science, Vol. 13, pp. 109-136, 1981.

[11] Giordana, and Saitta, L., "Modeling Production Rules by means of Predicate Transition Net-works," Information Science, Vol. 35, pp. 1-41, 1985.

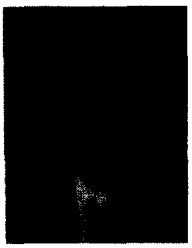
[12] Murata, T., and Zhang, D., "A Predicate-Transition Net Model for parallel Interpretation of Logic Programs," IEEE Trans. on SE., Vol. 14, No. 4, April, pp481-497, 988.

[13] Pederson, K. "Well-structured Knowledge Bases - Part I," AI Expert, April, pp. 44-55, 1989.

[14] Pederson, K. "Well-structured Knowledge Bases - Part II," AI Expert, July, pp. 45-48, 1989.

[15] Pederson, K. "Well-structured Knowledge Bases - Part III," AI Expert, November, pp. 36-41, 1989.

◐ 저 자 소 개 ◑



조 상 엽

1986년 한남대학교 전자계산학과(공학사)
1988년 중앙대학교 대학원 전자계산학과(이학석사)
1993년 중앙대학교 대학원 전자계산학과(공학박사)
1993~1994 중앙대학교 컴퓨터소프트웨어 연구소 객원연구원
1995~현재 청운대학교 인터넷컴퓨터학과 교수
관심분야 : 인공지능, 퍼지이론, 페트리네트 응용
E-mail : sycho@chungwoon.ac.kr